Vision based obstacle avoidance and motion tracking for autonomous behaviors in underwater vehicles

Marco Leonardi, Annette Stahl
Michele Gazzea
Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, NTNU AMOS, Trondheim, Norway.
Email: marco.leonardi@itk.ntnu.no, annette.stahl@ntnu.no, micheg@stud.ntnu.no

Martin Ludvigsen
Ida Rist-Christensen
Department of Marine Technology, Applied Underwater Robotics Lab, NTNU, Trondheim, Norway.
Email: martin.ludvigsen@ntnu.no, idaris@stud.ntnu.no

Stein M. Nornes
Department of Marine Technology, Centre for Autonomous Marine Operations and Systems, NTNU AMOS, Trondheim, Norway.
Email: stein.nornes@ntnu.no

***Abstract*—Performing reliable underwater localization and maneuvering of Remotely Operated underwater Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs) near nature protection areas, historical sites or other man-made structures is a difficult task. Traditionally, different sensing techniques are exploited with sonar being the most often used to extract depth information and to avoid obstacles. However, little has been published on complete control systems that utilize robotic vision for such underwater applications.**

This paper provides a proof of concept regarding a series of experiments investigating the use of stereo vision for underwater obstacle avoidance and position estimation. The test platform has been a ROV equipped with two industrial cameras and external light sources. Methods for underwater calibration, disparity map and 3D point cloud processing have been used, to obtain more reliable information about obstacles in front of the ROV. Results from laboratory research work and from field experiments demonstrate that underwater obstacle avoidance with stereo cameras is possible and can increase the autonomous capabilities of ROVs by providing appropriate information for navigation, path planning, safer missions and environment awareness.

Keywords*—*underwater stereo vision, obstacle avoidance, ROV, AUV, motion tracking, position estimation

I. Introduction

ROVs require a constant supervision during their missions and information about their environment. Even if underwater operations are characterized by low speeds, they can be very challenging due to the difficulty of not having a complete overview of the surroundings, due to the lack of light, due to non-linear distortions introduced by the water (relevant for visual sensors) and due to the impossibility to use high frequency radar because of high attenuation in salt water.
AUVs started to replace ROVs in several context, but due to limitations in the current artificial intelligence technology, their operative context is limited and ROVs are still broadly used today. ROV navigation is usually performed by one or more human operators that orient themselves with the help of a compass, a sonar and several cameras. By improving safety and success of the ROV missions a certain degree of autonomy within the ROV operations can be included. Cameras provide - compared with other sensors - cost effective information about the environment in high-resolution. That makes the implementation of such a system very interesting for obstacle avoidance, motion tracking, station keeping and drift correction.

A. Related Work

There are several sources in literature about obstacle avoidance and motion estimation utilizing vision for mobile and partial autonomous robots, but most of these studies are conducted for terrain, humanoid or aerial and not so much for underwater vehicles. An example of robotic vision and machine learning for autonomous underwater operations can be found in the field of visual servoing [1], [2].
Examples for underwater obstacle avoidance are in general based on multibeam echo-sounders [3] or other kind of sonars [4]. A previous work for underwater obstacle avoidance presents a structured light based visual subsystem, leading to the reconstruction of a 3D sea-line profile [5].
For motion estimation using cameras in the underwater environment the paper [6] can be considered. A collection of experiments is presented in [7] by testing a developed sensor fusion approach with a monocular camera and an Inertial Measurement Unit (IMU) [8]. Mosaicking techniques can also be exploited for motion estimation and are demonstrated in [9].

II. Hardware Setup

All field experiments where conducted using a ROV Sperre SUB-FIGHTER 30K (cf. Fig. 1). The ROV has been equipped with two Allied Vision Prosilica GC1380 cameras (CCD, global shutter, GigE Vision complaint, 1360x1024 pixels resolution), with a 8mm fixed focal length, an infinity focus and a maximum opening aperture. The cameras were placed inside two separate custom-made waterproof enclosures rated for 3000 meters depth. Two light sources (OSRAM 400W HMI) were utilized to illuminate the scene in front of the stereo system. Tests for obstacle avoidance and position estimation were performed at a depth of around 70 meters. The testing site we have chosen is the area around a shipwreck (“Hercules”, a fishing boat) in the Trondheimsfjorden. We have choosen a baseline of 9.5cm [10]. The cameras were synchronised using a software trigger and set with the standard factory

978-1-5090-5278-3/17/$31.00 ©2017 IEEE

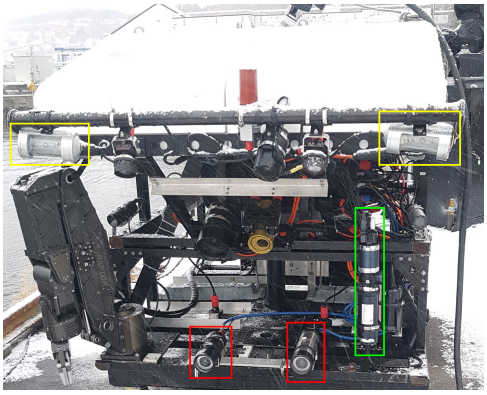

Fig. 1: ROV 30K: Stereo camera system (red rectangles), front looking sonar (green rectangle), light sources (yellow rectangles). See Fig. 2 for the final camera setup.

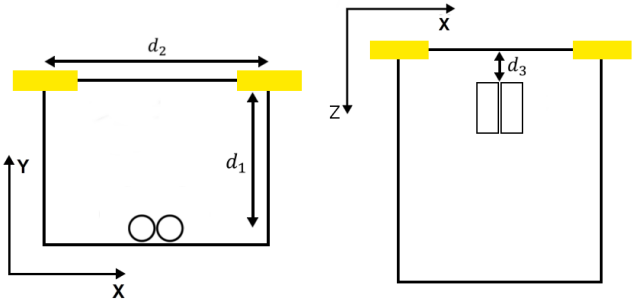

Fig. 2: Schematic drawing to illustrate the stereo camera - light source hardware setup at the ROV. **Left:** Front view, light sources (yellow), stereo camera (two circles), d_1=40cm and d_2=47cm. **Right:** Depth distance offset between the light sources and the cameras with d_3=20cm.

parameters except for the exposure time (15ms) and gain (15). In addition, we exploited a front looking sonar (Kongsberg Mesotech MS1000 with a high resolution head) for comparing the 3D information extracted from the stereo setup.
In the following we describe the tests we performed with respect to camera calibration, disparity map pre-processing, disparity map computation, 3D point cloud extraction and processing.

III. Camera Calibration

Camera calibration is a crucial step for extracting 3D information out of stereo images. Intrinsic and extrinsic camera parameters are computed along with a set of parameters that help to correct for lens distortions.
The intrinsic matrix contains the internal pinhole camera model parameters, and the extrinsic matrix describes the rotation and translation of the second camera with respect to the first camera, in a world coordinate frame. Typically, radial and tangential distortions are corrected. The radial distortion with parameters k_1, k_2, and k_3 can be described by the following equation

$$\begin{aligned} x_r &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_r &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ r^2 &= x^2 + y^2. \end{aligned}$$

Note that such distortions are commonly produced by standard lenses. The tangential distortion can be expressed as

$$\begin{aligned} x_t &= x + [2p_1 xy + p_2((x^2 + y^2)^2 + 2x^2)] \\ y_t &= y + [2p_2 xy + p_1((x^2 + y^2)^2 + 2y^2)] \end{aligned}$$

with parameters p_1 and p_2, that appear in case the lens is not mounted parallel to the sensor. The elements within the intrinsic camera matrix

$$K = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

are the horizontal focal length f_x, the vertical focal length f_y, a skew parameter s (set if the camera pixels are not perfectly squared) and the components x_0 and y_0 representing the principal point offset. The rotation matrix R along with the translation vector t represent the extrinsic camera parameters. There are many methods available for determining these unknowns and we refer to [11] for an introduction to multiple view geometry.
We exploit image rectification in order to simplify the correspondence problem (the problem of finding matching points between stereo image pairs). When the intrinsic and extrinsic matrices are computed the homographies needed for rectifying a pair of images can be calculated algebraically. The first step is to calculate the fundamental matrix F, a 3×3 matrix of rank 2 that satisfies the following equation:

$$p'^T F p = 0.$$

Here p and p' are corresponding points in homogeneous coordinates between two cameras, so that Fp represents an epipolar line on which the point p' must lie in the other image [11]. The Fundamental matrix F can be calculated directly using the intrinsic matrices K and K' of the cameras along with the extrinsic rotation R and translation t

$$F = K'^{-T} [t]_\times R K^{-1},$$

where the operator $[\cdot]_\times$ denotes a matrix that performs the vector cross product. After F is obtained it's possible to determine the epipoles by finding the left and right null spaces of F

$$e'^T F = 0, \qquad Fe = 0.$$

This can be done by the Singular Value Decomposition (SVD) of the matrix F as follows

$$F = UDV^\top,$$

where U and V are two orthogonal matrices and D is a diagonal matrix. The i-th diagonal element of D is defined as the i-th singular value of F and the i-th column of U and the i-th column of V are the corresponding left singular vector and right singular vector, respectively of F. The right null space of F corresponds then to the column vectors of V that belong to zero singular values. The left null space of F corresponds to the rows of $U^\top$ with singular values equal to zero.

For the following rectification step it is required to find a projective transformation H' that maps the epipole e' to the point at infinity $(1, 0, 0)^T$. Then the matching projective transformation H, which maps the remaining epipole e to infinity is computed. Note that this also minimizes the least-squares distance:

$$\sum_i d(Hx_i, H'x'_i).$$

The rectification of the images itself is performed by re-sampling the two images according to the respective projective transformation H and H'.

Calibration of a stereo setup is a well studied subject and several toolboxes for different programming environments are available. The toolboxes used during our studies are the MATLAB integrated toolbox [12] (cf. Fig. 3), Caltech toolbox [13] and calibration functions from OpenCV [14].

When working with computer vision approaches for underwater applications the low contrast of images [15] impacts

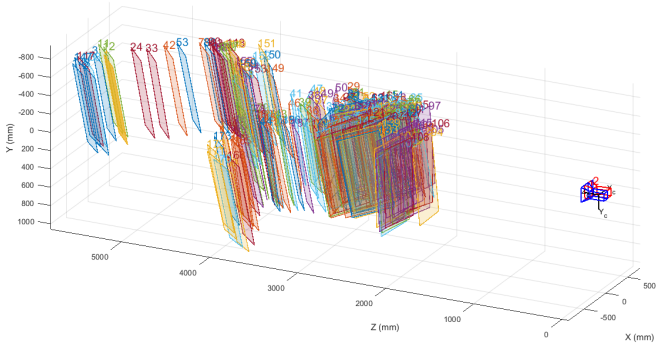

Fig. 3: Visualization of the reconstructed calibration patterns used in the stereo camera setup.

the ability to detect edges and corners, that is the first pre-processing step of many calibration toolboxes. Experiments showed that this problem is not significant and the automatic detection of the checkerboard worked well for images with reasonable quality (sharp enough and good illuminated, cf. Fig. 4). In order to assess the quality of the calibration results we

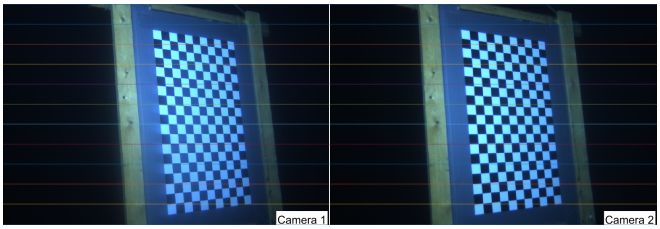

Fig. 4: Illustration of a rectification computed after calibration. Note that corresponding points are found on corresponding horizontal lines.

evaluated them in two steps, the first one was to compute the mean reprojection error. The reprojection error is defined as the distance in pixels between the actual projection of a calibration point in 3D and the reprojection of the reconstructed point onto the camera plane (cf. Fig. 5) using the 4×3 camera matrix $P = [R \mid t]^T K$. The reprojection error is mostly used

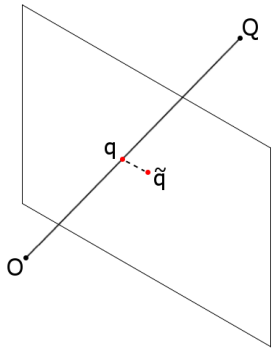

Fig. 5: Representation of the reprojection error: O camera origin, Q 3D point in space, q actual projection of the point Q on the camera plane, $\tilde{q}$ reprojected point. The reprojection distance is the euclidean distance in pixel between q and $\tilde{q}$.

to evaluate the result of a camera calibration, but at the same time it can be artificial lowered by iteratively removing images from the calibration images with the highest reprojection error (and it is also sensible to image resolution). As it may happen that the reprojection error is low, but the calibration is still sub-optimal (this occurs for example if the calibration pattern does not cover the full image [12]) we manually checked the rectified images for ensuring a good calibration.

The first tool that we tested for calibration was the integrated MATLAB toolbox. Two papers are cited in the toolbox documentation: The first one [16] describes a common method for camera calibration and is based on the identification of an arbitrary orientated planar pattern. The second paper [17] describes a calibration procedure, which consists of multiple steps: identification of a known pattern, a direct linear method for computing internal camera parameters that does not take into account lens distortion, followed by a Levenberg-Marquardt optimization in order to refine the results and calculate external parameters, taking into account the lens distortion correction. The calibration experiments were performed using two datasets: A dataset composed of 354 and a subset composed of 20 selected stereo-pair images. Using the integrated MATLAB toolbox we found a robust mean reprojection error of 1.08px and an average focal length of 1720.17px with an average estimation error of 0.09%. Removing all image pairs with a reprojection error of 2 pixels or more the average reprojection error reduces to 0.74px, the estimation error of the focal length reduces to 0.06% with an estimated focal length of 1719.36px. Note, that changing the optimization parameters did not impact theses results significantly.

The second calibration procedure that we tested was built from OpenCV calibration functions. OpenCV stereo calibration functions provide an automatic detection procedure of the checkerboard. By applying a histogram equalization to the gray value image and an adaptive threshold a binary image is generated. The corners of the checkerboard are detected with subpixel accuracy [18] using the respective OpenCV function (cornerSubPixel). A final optimization procedure is performed for estimating the lens distortions. In the used calibration example program the intrinsic camera matrices are estimated with

an initial guess. For optimization the Levenberg-Marquardt method is utilized with a termination criteria of 100 iterations or a delta change less than e^{-5}.
The calibration procedure using the Caltech calibration toolbox for a stereo setup starts with a manual independent calibration of the two cameras. For each camera the images have to be processed manually by providing the top left, top right, bottom left and bottom right corners of the calibration board. This process is not suited for processing a large amount of images, but allows to obtain a calibration when the automatic checkerboard detection fails due to non-optimal images (noise, non-homogenious illumination, etc.). The corners are then identified with subpixel accuracy using a Harris corner detector with a Gaussian mask.
The Caltech calibration toolbox estimates by default radial distortions (two coefficients) and the tangential distortions (two coefficients). The calibration process involves calibrating the two cameras separately before the stereo calibration procedure is called.
A satisfying rectification has been obtained with all three implementations. In order to evaluate the calibration results we compare the average reprojection error, the checkerboard recognition rate, the estimated focal length, the principal point, the rotation angle and the baseline for each calibration method.
In Table I and Table II f_x is the horizontal focal length, f_y is the vertical focal length, x_0 and y_0 are the coordinates of the principal point, x_d, y_d and z_d are the components of the translation vector, α, β and γ are the Euler angles around the x, y and z axis respectively.

TABLE I: Calibration results full dataset

Parameters	OpenCV	MATLAB
$avg\ f_x$	1739.67px	1720.18px
$avg\ f_y$	1739.67px	1719.87px
$avg\ x_0$	676.39px	677.17px
$avg\ y_0$	506.54px	512.14px
x_d	-96.78mm	-100.13mm
y_d	-2.05mm	-2.56mm
z_d	-2.04mm	-12.16mm
α	$-1.32°$	$1.25°$
β	$-0.89°$	$2.41°$
γ	$2.69°$	$-2.29°$
Reproj. Err.	2.11px	1.08px
Check. recog.	58 pairs	196 pairs

The first experiment (cf. Table I) shows that the programm using OpenCV functions provides a lower detection rate of the checkerboard. The performance of the calibration in the OpenCV implementation that we used, in terms of the re-projection error, is poorer, but the translational displacements of the second camera relative to the first are more similar to those that we have physically measured. The biggest difference between the two implementations in proportional terms is the z_d parameter, MATLAB shows that the second camera is 12mm behind the first one, which is not true.
For the Caltech calibration we generated a subset of 20 images, since manual selection of the corners was too time consuming for the entire dataset. In this calibration run (cf. Table II) it is possible to see that the MATLAB integrated toolbox and the Caltech toolbox tends to agree on many parameters, but again a strange behaviour in the offset z_d is present, and again all three implementations does not agree on the rotation angles, confirming that ideally a penalty term should be introduced during the parameter optimization forcing the angles close to zero. The tests that follow are performed using the MATLAB integrated calibration toolbox. We investigated the calibration results for different camera setups (cameras without enclosure, cameras inside the waterproof enclosures, and the stereo camera system underwater). The aim of these experiments was to determine which lens parameters are suitable for our specific application and to understand how these three different setups affect the calibration. Table III shows that the field of view decreases on average by 30% due to the refraction of water, but the combination of the frontal glass and water results in a decrease of the field of view of about 35%. This is a quite large change and has to be taken into account when working with cameras underwater.

TABLE II: Calibration results 20 stereo images

Parameters	OpenCV	MATLAB	Caltech
$avg\ f_x$	1683.73px	1715.78px	1721.29px
$avg\ f_y$	1683.73px	1718.26px	1721.54px
$avg\ x_0$	664.41px	699.70px	700.37px
$avg\ y_0$	511.13px	528.78px	520.00px
x_d	-102.16mm	-99.58mm	-101.34mm
y_d	2.07mm	-3.39mm	-1.77mm
z_d	1.67mm	6.81mm	-5.65mm
α	$2.49°$	$1.27°$	$-1.47°$
β	$-0.81°$	$2.31°$	$2.81°$
γ	$-1.34°$	$-3.06°$	$4.09°$
Re-proj. Err.	0.79px	0.75px	0.80px

TABLE III: MATLAB Calibration results

Params	Underwater	Out. w. encl	Out. w.out encl.
Re-proj. Error	1.08px	0.40px	0.33px
Focal Len. px	1720.17	1286.23	1244.00
Focal Len. mm	11.13	8.32	8.05
H. FOV	$43.14°$	$55.74°$	$57.62°$
V. FOV	$33.03°$	$43.27°$	$44.83°$

IV. Disparity map calculation

In order to successfully perform visual obstacle avoidance underwater one has to estimate the 3D world coordinates of objects detected in front of the ROV, therefore, the next step after calibration and image rectification is to extract 3D information from the stereo recordings. This implies finding correspondences between the stereo image pairs. A disparity map represents this information and is a matrix of the size of the rectified image, containing in each element the offset between corresponding pixels (generally from the left to the right image).

Many algorithms have been proposed in the literature to solve the correspondence problem [19]. These methods can be classified into dense and sparse reconstruction, local, semi-global and global approaches. For the purpose of robot navigation, where limited computational resources are available and a fast reaction time with respect to the constantly changing environment is required, it's better to focus on computationally effective approaches. The algorithms that have been tested for the disparity calculation are the classical block matching (local approach), a semi-global block matching [20] and a global graph cut based algorithm [21], [22]. An open-source implementation of the graph cut based algorithm can be found in [23].

Block matching algorithms for disparity estimation calculate a score for a pixel involving it's neighborhood (generally the area involved in the calculation of this score is a square, so the terminology "block") and finding the block in the other image that has the most similar score. The matching is conducted exploiting the epipolar constraint, so that correspondences are searched along corresponding horizontal lines and the matching block is the one with the highest similarity measure. The similarity measure itself has an impact on computational efficiency and matching quality.

In our experiments we tested a simple and fast pixel block similarity measure known as Sum of Absolute Differences (SAD). Given a pair of square blocks of pixels A and B, both of dimension $n \times n$, the similarity measure is determined as follows:

$$SAD(A,B) = \sum_{i=1}^{n}\sum_{j=1}^{n} |a_{i,j} - b_{i,j}|.$$

We used the SAD block matching function from MATLAB with default parameter setting.

The semi-global block-matching approach [20] exploits an entropy based matching cost. The idea is based on a pixel-wise matching of Mutual Information and approximation of a global 2D smoothness constraint. The following energy function is minimized in order to obtain the disparity:

$$\begin{aligned} E(D) &= \sum_{p} C(p, D_p) + \sum_{q \in N_p} P_1 T[|D_p - D_q| = 1] \\ &\quad + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1]. \end{aligned}$$

The first term is the sum of all pixel matching costs for the disparities of D. The second term adds a constant penalty P_1 for all pixels q in the neighborhood N_p of p, for which the disparity change is maximal 1 pixel. The third term adds a larger constant penalty P_2, for all larger disparity changes [20]. Such a global minimization is a NP-complete problem and the solution is approximated by aggregating matching costs in 1D from all directions equally. The computational complexity of the algorithm is linear in the number of pixels and disparity range, obtaining an overall accuracy similar to global methods.

The aim of the Kolmogorov and Zabih stereo matching algorithm [21] is to minimize a non-convex objective function exploiting graph cut techniques. The energy for a match represented by f is given as follows:

$$E(f) = E_{data}(f) + E_{occlusion}(f) + E_{smooth}(f) + E_{unique}(f).$$

Here the data term measures how well a matched pair fits, the occlusion term minimizes the number of occluded pixels, the smoothness term penalizes the non-regularity of the configuration and the last term enforces the uniqueness of the match.

Due to the lack of ground truth data the disparity map evaluation has been performed by manually estimating a disparity map. Three scenarios were selected: A scene without an object in front of the ROV, a scene with a fish at the Hercules site and a scene with the Hercules site occluded partially by a dust cloud. We estimated the disparity map by starting with a 19x19 window sized semi-global approach. To ensure a good quality for the disparity map and to refine the contour of the objects, stereo anaglyph spectacles were used.

As our goal is obstacle avoidance we first needed only a rough distance estimate for an object in front of the underwater robot, as this already allows to plan and perform object avoidance actions. However, false object detections (noise) should be restrained in order to prevent unnecessary actions. We used the following quality measures for evaluating the computed disparity map: Root Means Square Error (RMSE), the percentage of "corrent" matchings (within a 3 pixel range) and the percentage of noise detected in the image. We define noise as the presence of a false match.

The first batch of experiments (cf. Table IV) were performed with block matching, first the uniqueness threshold is kept at 15 and the window size is changed, then the window size is kept fixed, where the best result (in terms of RMSE and obstacle detection rate) was reached and others parameters are varied. The second batch of experiments (cf. Table V) were performed using the semi-global approach, which turned out to be sensitive only to the window size. The graph cut (cf. Table VI) approach has been run, where the penalty parameter K (evaluating occluding pixels), $\lambda 1$ and $\lambda 2$ (smoothness parameters) and edge threshold were automatically determined. The presented results are computed as the average over the three test scene images. The rectified image size is 1168x788 pixels and the CPU used for this experiment was a Intel Core i7-5820K. The best performance has been achieved with the SAD block matching, with

TABLE IV: SAD Block matching results

W. size	Uniq. T.	RMSE	Obstacle Detec.	Noise	Time
7x7	15	27.84	22.11%	7.21%	**0.026**s
19x19	15	15.60	57.11%	0.25%	0.027s
55x55	15	14.36	**67.45**%	0.20%	0.162s
149x149	15	14.60	31.98%	0.25%	0.280s
55x55	-	55.47	25.50%	55.83%	0.153s
55x55	8	**12.94**	58.98%	0.81%	0.161s
55x55	21	14.90	65.38%	0.07%	0.158s
55x55	55	16.17	33.33%	**0**%	0.153s

TABLE V: Semi-Global Block matching results

W. size	RMSE	Obstacle Detec.	Noise	Time
7x7	28.50	33.42%	25.32%	0.45s
19x19	30.74	33.16%	27.61%	0.45s
55x55	15.65	28.41%	0.14%	0.45s
89x89	23.48	25.93%	4.22%	0.46s
149x149	18.87	21.83%	3.39%	0.49s

TABLE VI: Graph cut results

K	$\lambda 1$	$\lambda 2$	Data cost	RMSE	Obs. Detec.	Noise	Time
78.13	58.11	19.4	L2	76.75	0.01%	33.9%	432.5s

respect to computation time and obstacle detection rate. The semi-global approach provides a slightly more dense disparity map, but is less accurate and slower. The graph cut approach turned out to be too slow for real time obstacle avoidance. Note, that the execution time of the SAD block matching with a fixed window size depends only on the images size and the used disparity range.

The generation of the disparity map can be improved by pre-processing and post-processing operations like gap filling. Common pre-processing procedures for disparity map enhancement are homomorphic filtering and histogram matching. Homomorphic filtering can be applied to remove the slow changing illumination within an image, while preserving the high frequency component of the reflectance. Assuming that the reflectance and illumination is multiplicative within the intensity one remove the slow changing illumination by computing the logarithm of the image and applying high-pass filter to the result. Applying the inverse of the logarithm leads to an enhanced image. Histogram matching is a process were the histograms of two images (disparity map) are made similar, to homogenize the images so that a simple and fast matching procedure like SAD performs well. The algorithm calculates the cumulative probability distribution (cpd) of the reference image and the target image. The two cpd's are then used to build a look up table in order to update the intensity values of the target image. We repeated the disparity map tests after filtering the rectified images with homomorphic filtering, histogram matching and the two combinations of these. We observed that filtering did not improve the disparity map, given the fact that our performance indexes are not based on a ground truth, but on a manually refined disparity map, the performance fluctuations were not significant.

V. 3D INFORMATION EXTRACTION AND PROCESSING

By using the camera parameters we compute for each pixel with known disparity its corresponding 3D coordinates. The first step is to calculate the depth (z-coord.)

$$Z_p = \frac{fb}{x_{lp} - x_{rp}}, \tag{1}$$

where f is the focal length, b the baseline, and $x_{lp} - x_{rp}$ the disparity, i.e. the correspondence difference between the left pixel and the right pixel of point p. Then x and y coordinates can be computed as follows:

$$X_p = \frac{x_p Z_p}{f}, \quad Y_p = \frac{y_p Z_p}{f}. \tag{2}$$

False matchings (noise) will also appear as points in the 3D point cloud. This noise could be removed already in the disparity map, however noise filtering in the point cloud leads to better results.

One possible way to filter the point cloud is by applying a simple statistical filter [24]:

$$P_d = \{p_i \in P_{raw} | \, \|p_i - p_j\| > \mu + d_{thresh} \ \sigma \ \},$$

where P_d is the denoised point cloud, μ and σ are the mean and standard deviation of the nearest neighbour distances. Another possibility is to filter noise by point clustering. A subsequent noise filter can be applied to remove clusters with a low number of points, unstructured distribution, etc. We define a cluster-density based hypothesis by: The number of clusters and their shapes are unknown and we suppose that the cluster density is a function of depth, given the increased uncertainty in the stereo 3D reconstruction due to quantization errors [25]. However, in the literature various clustering algorithms are known, like centroid-based and/or distribution-based clustering relying on the knowledge of number and/or shape of clusters. Another more flexible density-based clustering approach, that also incorporates the presence of noise, is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) approach [26]. The parameters used by this algorithm are the point cluster density and the minimum number of points that can build a cluster. We note that the number of clusters is determined automatically. Hierarchical clustering is also suitable for this filtering problem, and we tested therefore an algorithm called Ordering Points To Identify the Clustering Structure (OPTICS) [27]. It is based on DBSCAN and addresses it's major weakness: the inability of finding meaningful clusters in data of varying density. OPTICS achieve the ability of finding meaningful clusters in data of varying density by putting the points that are spatially close to each other in a points list. Using this list a reachability-plot (a special kind of dendrogram), a hierarchical structure of clusters can be obtained. Input parameters for OPTICS are the minimum number of points per cluster and a search radius ε to consider point-to-point distances.

We analyzed our cluster-density based hypothesis by extracting clusters from all images we had (calibration and field test datasets). The disparity maps have been generated by block matching with the SAD metric using a window size of 7x7 (leading to noisy disparity maps) and an uniqueness threshold of 15. In order to reach real time performance the point cloud has been uniformly downsampled to 10% of the original amount of points. Clusters have been extracted with OPTICS with a minimum amount of 10 points and ε=0.05. The cluster density is the average distance of the cluster points from it's centroid. We are interested in removing noise within the 3D point cloud that could affect the quality of the path planning. We are looking for a noise removal process that removes points that do not belong to obstacles and, at the same time, keeps all relevant points belonging to obstacles. In order to evaluate the noise removal, we defined two performance indexes, the first one is related to the ability to not remove relevant obstacle points and is defined as the percentage of retained points within 0.1m from the points extracted from the estimated disparity map. The second performance index is related to the noise removal and is defined as the amount of removed points that are 0.1m further away from the points extracted from the estimated disparity map.

In addition to the noise removal that we discussed we added a modified version of DBSCAN to the test group that exploits prior knowledge (a empirical determined function about the clusters density, without compromising it's running time). The modification of DBSCAN consists in changing the input parameter ε to a function of depth of the currently evaluated

point. Note, the function $regionQuery(P, \varepsilon)$ returns all the

Algorithm 1 $DBSCAN\ for\ Stereo\ Point\ Cloud$

```
1: procedure DBSCAN(D, MinPts)
2:     C = 0
3:     for all point P in dataset D do
4:         if P is visited then
5:             continue
6:         mark P as visited
7:         NeighborPts = regionQuery(P, f(P_z))
8:         if sizeof(NeighborPts) < minPts then
9:             mark P as noise
10:            continue
11:        else
12:            C = next cluster
13:            expCluster(P, NeighborPts, C, minPts)
```

Algorithm 2 $Subroutine:\ expandCluster$

```
1: procedure expCluster(P, NeighborPts, C, MinPts)
2:     add P to cluster C
3:     for all point P' in NeighborPts do
4:         if P' is not visited then
5:             mark P' as visited
6:             NeighborPts' = regionQuery(P', f(P'_z))
7:             if sizeof(NeighborPts') >= minPts then
8:                 NeighborPts += NeighborPts'
9:         if P' is not yet member of any cluster then
10:            add P' to cluster C
```

points with the euclidean distance smaller than ε from the point P including the point P itself. $f(P_z)$ is the empirical determined function that follows the found average cluster density in our experimental cluster density analysis.

Following we present an evaluation in terms of kept object points and correct noise removal of the discussed filtering approaches.

The point clouds are obtained from disparity maps generated with block matching and SAD metric. A set of 3 different window sizes (7x7, 19x19 and 55x55) and an uniqueness threshold of 15 was used to create the point clouds. The evaluation include DBSCAN ($\varepsilon = 0.15$, $minPts = 200$), OPTICS (same settings as DBSCAN), our modified DBSCAN ($minPts = 200$) and a statistical outlier removal approach [24] that evaluates 30 neighbors with $d_{thresh} = 1$ by applying a simple statistical filter. The results in Table VII show that

TABLE VII: Point Cloud Noise Filtering Results

Filter	Obj. Preservation	Noise Removal
DBSCAN	73.6%	89.5%
DBSCAN Stereo	96.3%	70.1%
OPTICS	53.7%	87.5%
Statistical	99.7%	45.2%

the performance of DBSCAN can be improved by using prior knowledge about how the cluster density changes (with a small cost of the noise removal). The statistical approach has shown great potential by preserving points belonging to the object, but it removes the least amount of noise. OPTICS does not perform well as it erodes the obstacles points. Hierarchical clustering assumes a hierarchical clusters structure (organization) that not always makes sense in this kind of clusters, and so the performance of OPTICS is the worst of the set regarding the preservation rate.

VI. Motion Tracking and Estimation

The stereo recordings were utilized to estimate the motion of the ROV by tracking features over consecutive image frames. Features from two consecutive images are first extracted, matched and then their 3D position is computed. An optimization algorithm provides the transformation that occurred between the image pair.

Features are extracted from the current stereo image pair (*currL*, *currR*) at time T_1 in addition to the features computed for the image pair (*prevL*, *prevR*) from the previous time $T_0 = T_1 - \Delta T$.

Features are detected in the images separately with the rotation and scale invariant Speeded Up Robust Features (SURF) descriptor [28] and then matched together spatially and temporally. Matching is performed using feature descriptors extracted from SURF and comparing their norm using the Sum of Squared Differences (SSD). We identify features that are present in all four images. These are then used to estimate the motion.

Let's denote with $\mathcal{F}_f$ the location of the features in frame f. The difference between matchings of features in frame $currL$ and frame $prevL$ is defined as

$$(du, dv) = \mathcal{F}_{currL} - \mathcal{F}_{prevL}. \tag{3}$$

Once we get such sets, we can use standard stereo reconstruction methods to estimate the three dimensional position of the features.

Given the sets of matched features found in the previous step, one can find the transformation (rotation and translation) which best describes the relation between the features at time T_0 and at time T_1. This can be formulated as an optimization problem (cf. also Figure 6) where one aims to find the minimum of an objective function defined as

$$f : \mathbb{R}^6 \to \mathbb{R} \qquad f(\underbrace{x, y, z, \theta, \alpha, \psi}_{\text{variables}}, \underbrace{P_{T_1}, \Pi, u_0, v_0}_{\text{given values}}) \mapsto \varepsilon, \tag{4}$$

where $\mathbf{x} = (x, y, z, \theta, \alpha, \psi)$ represents the motion (translation and rotation) of the camera between two frames at time T_0 and T_1, P_{T_1} are the 3D positions all the features detected at time T_1, $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection matrix used to map 3D coordinates onto the image plane. The variables (u_0, v_0) are the positions at time T_0 of all the features in the image plane. The median error of all N feature points is defined as

$$\varepsilon = \underset{i=1\ldots N}{\text{median}}\{(u_{0_i} - u_{e_i})^2 + (v_{0_i} - v_{e_i})^2\} \tag{5}$$

where (u_{e_i}, v_{e_i}) are the positions at time T_0 of the estimated features in the image plane as described in Figure 6. The optimal value is obtained as

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^6}{\arg\min}\, f(\mathbf{x}; P_{T_1}, \Pi, u_0, v_0). \tag{6}$$

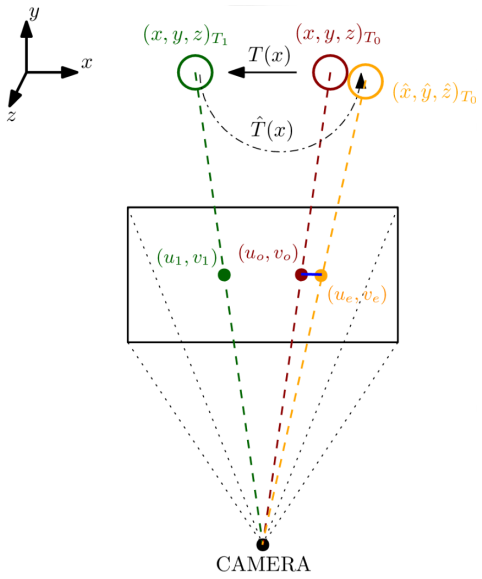

Fig. 6: Illustration of the minimisation problem (4). Example: Translation along the x-axis. Estimate the transformation ($T(x)$) between two frames, where only the 3D position of the features at time T_1 (green) and the position of the point in the image plane at both times T_0 and T_1 are known. We estimate the 3D position of the feature at time T_0 (orange) through the guessed transformation $\hat{T}$ and project it back to the image plane obtaining (u_{e_i}, v_{e_i}). The desired value of x is the one that minimises the distance between the two projections (blue segment).

To perform such a minimisation the *Nelder-Mead simplex method* has been used. It is a non-linear unconstrained model-free optimization procedure. The minimisation algorithm creates a polytope (simplex) on the variables space whose vertices sample the function at different locations, this simplex either expand or shrink in order to find the minimum.
The algorithm for estimating the camera motion between two frames is outlined in Algorithm 3.

We implemented two versions of the algorithm. The first one we call *(OME3)* [1] and it estimates only the translation and receives the differential angles as input while the other *(OME6)*[2] estimates all the six variables simultaneously ([29]) (cf. Fig. 9). The reason for that is, in some cases (especially with few features) the algorithm computes a transformation which is not the desired one but still explains the projected features quite well between the images. Note that equation (4) represents a non-convex optimization.

The camera pose (motion) estimation can be very challenging in some situations where the algorithm provides only a sub-optimal solution.

During lab experiments outside of the water we have seen that single (sparse) feature mismatching occurs regularly like shown in Figure 7.

[1]Optimised Motion Estimation - 3 parameters
[2]Optimised Motion Estimation - 6 parameters

Algorithm 3 Visual motion estimation

1: Get the stereo image pairs of 2 consecutive frames
2: Extract features
3: Match features in left and right images (spatial match)
4: Match them with the ones at previous step (temporal match)
5: Reconstruct these points which have been both spatially and temporally matched
6: Determine the camera transformation using Algorithm 4
7: Proceed to next frame and return to step 1

Algorithm 4 Pose estimation function

1: Use the previous time step's differential pose estimation as an initial guess
2: Build transformation matrix from previous two current frames
3: **for** $i = 1 \dots N$ (N number of matched features) **do**
4: Solve $\hat{P}_{T_0} = \mathbf{T} P_{T_1}$
5: Project $\hat{P}_{T_0}$ to image plane to get (u_e, v_e)
6: Compute the error ε as defined in (5)
7: Use Nelder-Mead method to update angles and translations
8: **if** Error is sufficiently low **then** variables have been estimated correctly and thus the algorithm can stop
9: **else**
10: Proceed to next frame and return to step 2

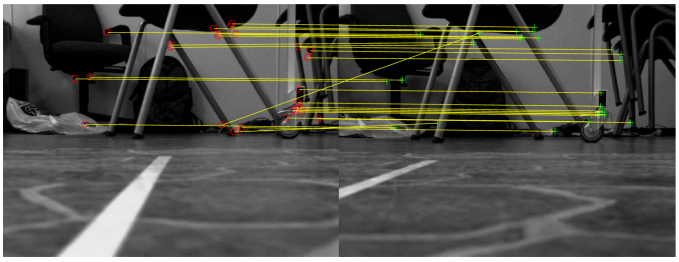

Fig. 7: Feature mismatching between left and right camera

Although such matching errors happen relatively due to the fact that features are tracked both temporally and spatially, they are still present in certain situations. Mainly when the environmental light is not good or texture regions look quite similar.

Such mismatchings can lead to an erroneous transformation estimation. However, since the objective function involves the median of the reprojection errors, outliers are not weighted. In Figure 8 features are drawn as red dots. The yellow lines indicate the larger matching positions in the previous frame.
For estimating the motion (between 2 frames) we exploited the rotation information provided by the IMU and estimated the differential translation with OME3. This result was refined by a following complete search finding all six variables (translation and rotation). From the obtained path in Figure 10 we can see that the estimated motion path agrees to a certain extend with the measured motion path. Over time the error accumulates and introduces a small drift. In all plots the blue line is the real trajectory while the red one is the estimated one.

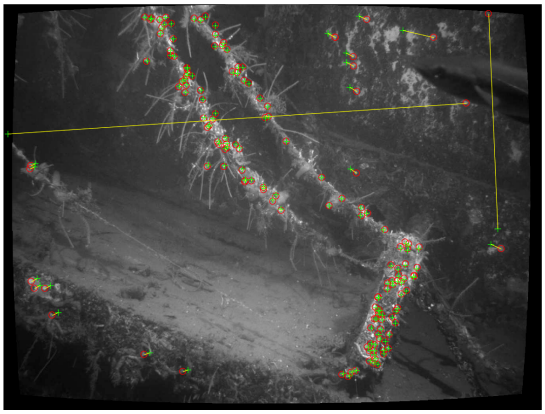

Fig. 8: Hercules site in Trondheimsfjorden: Feature matching result obtained in real time. Difference between real features (red) and the ones computed with the estimated transformation (green).

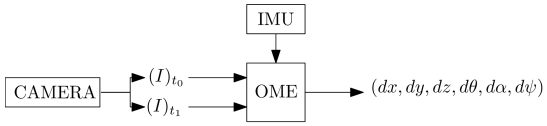

Fig. 9: Estimation algorithm running in mode (OME6)

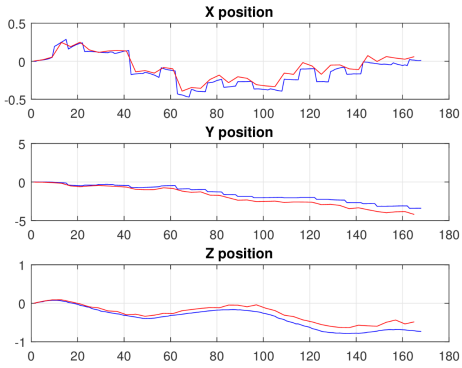

Fig. 10: Comparison of the estimated reconstructed trajectory (seconds, meters)

VII. Control System for Collision/Obstacle Avoidance

The stereo vision based collision avoidance system was implemented as a reactive part of an autonomy layer in the mission control system used for the autonomous ROV intervention [30]. The autonomy layer contains a deliberative module switching between several predefined behaviors in parallel with a system of reactive behaviors taking control of the vehicle in case of unwanted and unforeseen events, like an obstacle in the path of the vehicle. The system provides guidance input to the ROV control system. In our experiment, the vehicle was sent towards an obstacle known to the operator, in order to purposely test the detection of obstacles and to provoke reactive behavior for obstacle avoidance using robotic vision. When the obstacle in the point cloud (non empty), the orientation of the vehicle, and estimated distance to the obstacle is sent to the autonomy layer through User Datagram Protocol (UDP) communication. As a result, the reactive behaviour demands a change in heading of the vehicle. The vehicle turns until the obstacle is no longer detected in the stereo images, and a new way-point is defined straight ahead (cf. Fig. 11). During the

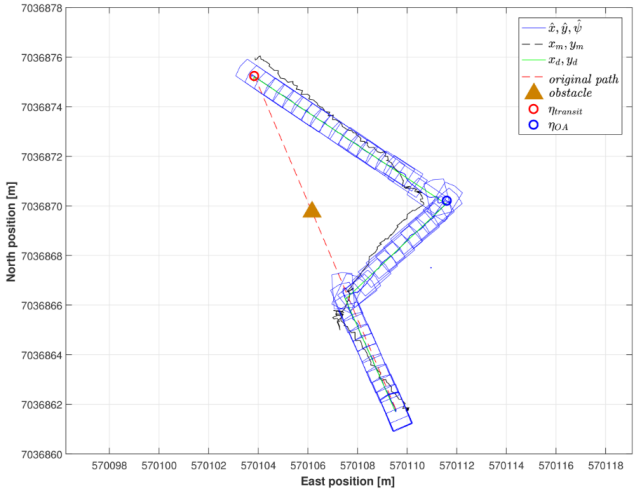

Fig. 11: ROV position in xy-plane during collision avoidance.

field testes it was confirmed that the 3D information coming from the stereo system were consistent with the information provided by the front looking sonar of the ROV. Note, fish were identified as obstacles. This problem can be addressed by a time analysis, clustering or machine learning approaches. Using MATLAB running on a i7-3820QM we were able to evaluate 5 point clouds per second, thus, given the slow speed of the underwater vehicle, we were able to achieve real-time performance. Real-time performance can be improved by reducing the resolution of the images in order to gain a higher frame rate.

VIII. Conclusion and Future Work

In this paper we presented a computer vision based ROV steering module that allows to add a certain degree of autonomy to a ROV mission. In particular we developed and analysed an obstacle avoidance module that is based on stereo-vision. It allows to detected obstacles in real-time and also determines the distance of the ROV to the object. Factors like turbidity and more difficult lighting conditions make the analysis of underwater video recordings in general more challenging than the analysis of video recordings in the air. Therefore we started with an analysis of three standard calibration implementations that are known to work well in air and exploited these for underwater stereo camera calibration. We found that the overall quality of the calibrations is lower comparing to calibrating stereo cameras in air but that all tested approaches in principle can be used to estimate the intrinsic and extrinsic parameters of the camera system. The task of computing a dense disparity map turned also out to be more challenging due to the lower quality of the calibration and due to the low sharpness of underwater images. However, fine tuning of a block matching approach with SAD has proven to be accurate

enough. Subsequent filtering of the extracted 3D point cloud of the scene has been achieved without any prior information by the means of statistical techniques and clustering methods, with the latter providing better results. Several tests with a ROV in the Trondheimsfjorden showed solid performance of the autonomous tracking and obstacle avoidance based on the stereo vision analysis even in absence of natural light. Future plans include the improvement of the underwater disparity map calculation and promising approaches are likely based on convolutional neural networks as they show for generic scenes a performance superior to other methods [31]. However, for a more accurate analysis and comparison of underwater disparity map calculations it is desirable to build an underwater ground truth data set for disparity maps.

ACKNOWLEDGMENT

This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU.

REFERENCES

[1] A. Carrera, N. Palomeras, D. Ribas, P. Kormushev, and M. Carreras, "An intervention-auv learns how to perform an underwater valve turning," in *OCEANS 2014-TAIPEI*. IEEE, 2014, pp. 1–7.

[2] P. Cieslak, P. Ridao, and M. Giergiel, "Autonomous underwater panel operation by girona500 uvms: A practical approach to autonomous underwater manipulation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 529–536.

[3] Y. Petillot, I. T. Ruiz, and D. M. Lane, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, pp. 240–251, 2001.

[4] E. O. Belcher, W. L. Fox, and W. H. Hanot, "Dual-frequency acoustic camera: a candidate for an obstacle avoidance, gap-filler, and identification sensor for untethered underwater vehicles," in *OCEANS'02 MTS/IEEE*, vol. 4. IEEE, 2002, pp. 2124–2128.

[5] G. Antonelli, S. Chiaverini, R. Finotello, and R. Schiavon, "Real-time path planning and obstacle avoidance for rais: an autonomous underwater vehicle," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, pp. 216–227, 2001.

[6] J. Evans, P. Redmond, C. Plakas, K. Hamilton, and D. Lane, "Autonomous docking for intervention-auvs using sonar and video-based real-time 3d pose estimation," in *Oceans 2003. Proceedings*, vol. 4. IEEE, 2003, pp. 2201–2210.

[7] P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu, "Experiments with underwater robot localization and tracking," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 4556–4561.

[8] L. Armesto, J. Tornero, and M. Vincze, "Fast ego-motion estimation with multi-rate fusion of inertial and vision," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 577–589, 2007.

[9] R. Garcia, J. Batlle, X. Cufi, and J. Amat, "Positioning an underwater vehicle through image mosaicking," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 2779–2784.

[10] J. Bercovitz, "Image-side perspective and stereoscopy," in *Photonics West'98 Electronic Imaging*. International Society for Optics and Photonics, 1998, pp. 288–298.

[11] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] "Matlab stereo calibration app," https://se.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html, accessed: 2017-01-11.

[13] "Matlab camera caltech calibration toolbox ," http://www.vision.caltech.edu/bouguetj/calib_doc/, accessed: 2017-01-11.

[14] "Opencv calibration toolbox," http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, accessed: 2017-01-11.

[15] M. Bryant, D. Wettergreen, S. Abdallah, A. Zelinsky *et al.*, "Robust camera calibration for an autonomous underwater vehicle," in *Proc. Australian Conf. on Robotics and Autom.* Citeseer, 2000, pp. 111–116.

[16] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[17] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Computer Vision and Pattern Recognition*. IEEE, 1997, pp. 1106–1112.

[18] R. Henkel, "Fast stereo vision with subpixel-precision," in *Proceedings of the sixth international conference on computer vision, Bombay*, 1998, pp. 1024–1028.

[19] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, vol. 2016, 2015.

[20] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Computer Vision and Pattern Recognition. CVPR 2005.*, vol. 2. IEEE, 2005, pp. 807–814.

[21] V. Kolmogorov, P. Monasse, and P. Tan, "Kolmogorov and zabihs graph cuts stereo matching algorithm," *Image Processing On Line*, vol. 4, pp. 220–251, 2014.

[22] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 508–515.

[23] "Kolmogorov and zabihs graph cuts stereo matching algorithm implmentation," https://github.com/pmonasse/disparity-with-graph-cuts, accessed: 2017-03-14.

[24] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[25] R. Balasubramanian, S. Das, S. Udayabaskaran, and K. Swaminathan, "Quantization error in stereo imaging systems," *International journal of computer mathematics*, vol. 79, no. 6, pp. 671–691, 2002.

[26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[27] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[28] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[29] M. Dunbabin, K. Usher, and P. Corke, *Visual Motion Estimation for an Autonomous Underwater Reef Monitoring Robot*. Springer Berlin Heidelberg, 2006, pp. 31–42.

[30] T. O. Fossum, M. Ludvigsen, S. M. Nornes, I. Rist-Christensen, and L. Brusletto, "Autonomous robotic intervention using rov: An experimental approach," in *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1–6.

[31] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1592–1599.